# *Introduction to Computational Physics*

*PHYS 250 (Autumn 2018) – Lecture 1*

David Miller

Department of Physics and the Enrico Fermi Institute
University of Chicago

September 27, 2018

## *Outline*

## *Computational Physics (PHYS 250)*

**Course Description PHYS 250 (link to Course Catalog)**

This course introduces the use of computers in the physical sciences. After an introduction to programming basics, we cover numerical solutions to fundamental types of problems, including cellular automatons, artificial neural networks, computer simulations of complex systems, and finite element analysis. Additional topics may include an introduction to graphical programming, with applications to data acquisition and device control.

**There are an infinite number of paths that we might follow and still not deviate from this description. I therefore would like to lay out some of the principles that will guide me, and us, in how we navigate through those many possibilities.**

## *Overarching Learning Goals for this Quarter*

- **Identify problems** that benefit from or require **computational/numerical approaches** and tools to solve.
- Develop an **algorithmic approach** to answering those problems computationally.
- Familiarity with **common computational algorithms** and numerical approaches to typical problems.
- Use **modern, high-level programming languages** to implement the computational algorithms.
- Use modern software tools for **developing, preserving, disseminating, and expanding** on those solutions.
- **Apply computational tools** to contemporary physics questions.

## *What are we doing here?*

*"The computer is incredibly fast, accurate, and stupid. Man is incredibly slow, inaccurate, and brilliant. The marriage of the two is a force beyond calculation."*

→ Maybe this is a quote from Albert Einstein, maybe it is from Leo Cherne...I'm not sure, but it's certainly true.

## *What are we doing here?*

## *Hello world!*

Code example

```
import numpy as np
import matplotlib.pyplot as plt

def p(x):
    return np.exp(-x**2)

#let's plot it
x = np.linspace(-3,3,100)
y = p(x)
plt.plot(x,y)
```

## *Hello world!*

```
import numpy as np
import matplotlib.pyplot as plt

def p(x):
    return np.exp(-x**2)

#let's plot it
x = np.linspace(-3,3,100)
y = p(x)
plt.plot(x,y)
```

## Outline

## *Conclusions*

- Something